4.1. Haz una copia del proyecto del ejercicio 3.5 de temas anteriores y pasa toda la lógica de negocio(todos los cálculos) a una capa de servicio. Puede ser una sola clase(CalculosService).
Hay proyecto.

4.2. Haz una copia del proyecto anterior y crea un área nueva para trabajar con fechas (nuevo controlador, nuevas vistas y nuevo servicio) con las siguientes características:
a) La url base de esta parte será /fechas.
b) Si le pasamos en la parte query una fecha, mostrará los días transcurridos desde el 1 de enero del mismo año. Las fechas se pasan siempre en formato YYYY-MM-DD.
c) Si le pasamos en la parte query dos fechas, mostrará los días comprendidos entre ambas fechas.
d) Si no se pasan ninguna fecha, mostrará los días transcurridos entre el 1 de enero y hoy.
e) Si pasamos /bisiesto/fecha1 mostrará si fecha1 pasada en el path es de un año bisiesto.
f) Si le pasamos /bisiesto/año1/año2 mostrará los años bisiestos comprendidos entre ambos años.
Puedes crear en el index unos enlaces a las URL creadas con distintas fechas como parámetro y probar de forma sencilla los casos anteriores.
No entiendo bien a que se refiere con eso de crea un área nueva. Imagino que es algo tipo en Netbeans con los paquetes, y que crearas una carpeta nueva, o algo del estilo desde VSc. Quizás es añadir otro @controler en el mismo archivo. Yo quise meter 2 llamados igual. Pero eso, lo intenté, porque no consigo meter 2 controladores llamados controller (podría llamarlos diferente también pero, en mi opinión, un controlador tiene que llamarse controller, aunque no creo que sea 100% necesario, mientras tengan su etiqueta @controller).
4.3. Haz una copia del proyecto anterior y crea una interfaz CalculosService pasando la clase anterior a llamarse CalculosServiceImpl y lo mismo con el servicio de cálculo de fechas.
No tengo el ejercicio anterior, así que no puedo hacer este.

4.4. Haz una copia del proyecto 3.4 pasando la lógica de negocio a una capa de servicio que contenga
la colección con los números aleatorios y los métodos para añadir nuevos números, eliminar números existentes, devolver los elementos de la colección y devolver la cantidad de elementos de la colección. Una vez que funcione la aplicación, prueba a ejecutarla en distintos navegadores a la vez. Explica en el PDF de soluciones de este tema qué ocurre, por qué y cómo solucionarlo. Para responder a estas últimas cuestiones vuelve al tema 1 a la sección de Scopes.

Ya hice el ejercicio y, aún no miré ese apartado de Scopes, lo miro después de escribir esto que creo, a ver si coincide o no. Pasa algo que, aunque no nos lo parezca a simple vista, es muy lógico. Ambos navegadores están mostrando y actualizando el mismo Html. Aunque no se nos muestre en cada uno de los navegadores hasta pulsar el botón para hacer una petición, si se está actualizando el mismo archivo, así que, si añadimos algo desde otro navegador/otra pestaña, no se nos muestra en la primera, puesto que no actualizamos esa

vista hasta hacer una petición desde ese navegador/pestaña. Por lo que, al actualizar la vista, lo hacemos desde la última versión actualizada, estuviéramos visualizandola o no. Habría sólo una solución (que yo encuentre), aunque no se ni si es posible. Que al actualizar la lista, se mande un “aviso” a todas las aplicaciones abiertas, de alguna forma que no sé, y que se actualice la vista en todas las apps abiertas cada vez que se ejecuta un cambio. Ya habiendo visto el apartado de requests del tema 1, es lo que dije antes. Al aplicar la recarga de la página, se crea una instancia nueva del objeto, porlo que, los cambios que hayan sido aplicados por otros usuarios se reflejan en esa nueva instancia, además del cambio que hayas pedido tú esa vez.